ED 312 998 IR 014 036

AUTHOR	de Diana, I. P. F.; Vos, Hans J.
TITLE	Abstract Representation of Tutorial CAI and the Development of an Adjustable Tutorial System.
INSTITUTION	Twente Univ., Enschede (Netherlands). Dept. of Education.
REPORT NO	ISBN-90-365-0087-7
PUB DATE	87
NOTE	40p.
AVAILABLE FROM	The Department of Education of the University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.
PUB TYPE	Information Analyses (070) -- Reports - Research/Technical (143)
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	Communication (Thought Transfer); *Computer Assisted Instruction; *Courseware; Models; *Systems Approach
IDENTIFIERS	*General Systems Theory; *Software Development

ABSTRACT

This paper presents a model of an abstract tutorial system. The first section discusses some of the problems encountered in producing courseware and examines the availability of software tools to support effective communication on three levels: (1) accuracy of communication; (2) transmission of the desired meaning; and (3) affecting conduct in a desired way. In the next section, the use of General Systems Theory (GST) as a framework for modelling is discussed. The third section introduces some elementary systems, and the fourth section addresses--on an abstract level--the use of pointers and the subject matter represented by a collection of four elementary frames. In the final section, the model is formalized using GST. Twenty-five references are listed. A list of the system variables and equations of the GST model are appended. (MES)

Abstract representation of tutorial CAI and the development of an adjustible tutorial system

I. de Diana
H Vos

Department of Education

University of Twente

ABSTRACT REPRESENTATION OF TUTORIAL CAI AND THE DEVELOPMENT OF AN ADJUSTABLE TUTORIAL SYSTEM

I. De Diana, H. Vos.

University of Twente, Enschede, the Netherlands.

Abstract representation of tutorial CAI and the
development of an adjustable tutorial system
I.P.F. De Diana, H.J. Vos.
Enschede : University of Twente,
Department of Education, 1987
ISBN 90-365-0087-7

Trefwoorden:
tutorial CAI
general systems theory
instructional systems modelling
courseware design
CAD/CAI
information technology

Colofon:

Typewerk	: E. Wigbold
Vormgeving omslag	: AV-sectie TOLAB
Reproductie	: Centrale Reproductie-afdeling Universiteit Twente
Oplage	: 50 exemplaren
Besteladres	: Universiteit Twente Mediatheek Toepepaste Onderwijskunde Postbus 217 7500 AE ENSCHEDE tel.: 053-893588

TABLE OF CONTENTS

ABSTRACT

An obvious problem in the construction of courseware for computer assisted instruction is the lack of analytical tools for testing the likely effectiveness of the courseware under construction.
The CAD-CAI research project of the University of Twente aims at the development of a set of analytical tools for courseware developers, integrated in a production environment. A first step in the development of such tools is perceived to lay in the modelling of the product type. The general systems theory has been used as a framework for modelling the tutorial instructional process. An unambiguous representation was developed, which served as the foundation for the realization of an adjustable tutorial "machine". This machine can operate as the execution mechanism for tutorial courseware that has been developed in the form of a network of frames (nodes). Thus both a general model has been constructed, based upon which specific tutorial courseware can be executed and a basis has been laid for manipulating various design options in order to be able to trace the likely effects of design choices. Design options in respect to decisionrules for steering the instructional process in the context of system controlled CAI.

1. INTRODUCTION

The production of courseware for computer assisted instruction (CAI) is regarded to be a complex and costly process. Various sources (Moonen and Gastkemper 1983, Kearsley 1983) report production time/on-line instruction time ratios varying between 80 and 200 (in hours).
The production of courseware is a complex activity because it is a combined process of developing computer software and an instructional system. Albeit an abundance of literature has been published both about software engineering and about the development of instructional systems, not quite much has been published about the engineering of courseware some of the exceptions being (Bork 1981, 1984).

Even though special instruments have been developed to support the development of courseware such as authoring languages and authoring systems, a fundamental problem is encountered during the development of courseware. It is more or less consciously realized by every developer of courseware, that no special instruments or methods exist by means of which the likely performance of courseware is analyzable during design.
As a consequence it is simply not possible for a designer to calculate the likely effects of various design options in terms of the expected likely performance of the courseware. Design choices are based on educated guesses and on trial and error based learning experiences gained from the development praxis.

Gaining access to a manipulatable model of the courseware system under design would provide the developer with a basis for the use of special facilities to experimentally test the effects of implementing various design options. Such a model in combination with attached analytical instruments would offer the developer the possibility to systematically alter parameters of the system and to calculate the results in terms of the expected performance of the courseware.

The CAD-CAI researchproject, described to some extent in this article, embodies an attempt to create a workenvironment for the development of courseware (called the EDUC system) in which the effects of various design options during the design of tutorial CAI can be analyzed. As the

construction of models is based upon the creation of an abstract representation of the system under study, the researchers started with the development of an abstract tutorial system. This abstract system was transformed into an adjustable tutorial software machine, forming a part of the before mentioned experimental workenvironment.
To this workenvironment software instruments for analyzing the effects of various design options have been attached.

2. WHY A WORKENVIRONMENT FOR THE PRODUCTION OF TUTORIAL COURSEWARE?

As CAI is based on the executior of computerprogrammes that carry out the instructional activities, these programmes should of course be free from syntactic errors. It is a well known fact that the design and construction of large computerprogrammes is a notorious activity in terms of costs in time and effort involved.

Producing courseware however means the production of an instructional system as well. The design and development of an adequate instructional system requires hardly less effort than the development of a larger scale computerprogram. It is an activity that needs carefull planning and a thorough knowledge of the instructional variables involved (Romiszowski 1982, 1984) Furthermore, during the development of courseware explicit attention needs to be given to the man-machine interface, being the various ways the human and the computer will interact during the execution of the instructional computerprogram (Kearsley and Hillelsohn 1982).

These three areas, computerprogramming, the development of an instructional system and the development of a man-machine interface mingle during the design and development process of courseware. Together they form a difficult mix from the perspective of an attempt to produce quality courseware at acceptable cost.

Several indications in the literature point out that the production of courseware is a highly time consuming activity as wel, the ratios of production time to net connect instruction time vary between 100 and 200 (in hours), depending mostly upon the difficulty of the instructional problem involved, the expertise of the developers and the adequacy of the hardware and software tools used (Moonen and Gastkemper 1983, Kearaley 1983).

It has been often noted in the literature that multi-disciplinary expertise (e.g. computer specialist, educationalist, graphics designer, psychologist) is needed for the production of quality courseware (for instance (Kontos, 1984). Forming (and keeping..) viable multidisciplinary work units often appears to be a difficult task (Francis 1979).

Keeping these problems in mind, it is no wonder that the history of the development of support tools for the production of courseware dates back as far as 1960 (Kearsley 1982) By means of these tools it was hoped that the production of courseware would become easier, that productivity could be increased and that the portability of courseware could be augmented by using back-end compilers to translate the code produced by these tools for different computersystems and to create a workenvironment for multi-disciplinary specialists. In general, these tools are divided in two classes, viz. authoring languages, being special purpose higher order application languages which facilitate the production of courseware (for a review see (Barker and Singh, 1982)) and authoring systems, being more or less integrated sets of software tools intended to make the production of courseware an activity for which no programming expertise is needed (Kearsley 1982).

Even though some indications in the literature can be found that (especially) authoring systems speed up the process of producing courseware (Avner 1979, Fairweather and O'Neal 1984) the production time and effort involved stays rather high. Taking it for granted that designing and developing quality instructional materials will remain an exacting activity, we rather pose the question why the effort involved quite often does not stand up to the resulting product quality.

A point of entry for answering this question can be found in the realization that computer assisted instruction is a form of communication, taking place between a source or sender of data (the computer and in last instance the producer of courseware) and a receiver of data (the human learner). Shannon and Weaver (1949) have pointed out, that three levels are involved in the communication of data, as far as the effectiveness of the message sent is concerned.

Level 1 pertains to the accuracy with which the symbols of communication are transmitted and received. Level 2 pertains to the precision with which the symbols of communication convey the meaning intended by the sender (in our case the sender is the author of the courseware). Level 3 pertains to the effectiveness with which the received message effects conduct of the receiver in the way desired by the sender. These levels build upon each other. Accurate reception of the symbols of communication is a pre-condition for the convey of the meaning encoded in the symbols. Reception of the meaning of the communication in its turn is a pre-condition for effecting the receiver's behavior in a desired way.

Returning to the question why the effort involved in the production of courseware quite often does not stand up to the resulting product performance, at least two factors seem to be involved. The production technology for courseware is a not well developed field, to say the least. The production of courseware is still an activity based primarily upon experiences gained in the developmental praxis. A typical developer goes through a period of some kind of apprenticeship, during which he typically learns to work with some authoring facility and learns a set of action rules, based upon which the production of courseware is carried out. Design choices during the production of courseware are mostly based upon educated guesses, combined with knowledge gained from trial and error based experiences in the development praxis, rather than on a systematic application of proven design knowledge. A fairly typical situation for a developing technology.
Furthermore, the developer of courseware is dependent on the adequacy and quality of the tools he is using. The availability of tools that offer some support for the realization of effective communication on the three levels mentioned therefore seems an important factor.
Looking at available software tools for courseware-producers from the perspective of the support they offer for the production of effective courseware , we obtain the following picture.

Level 1. The accuracy of communication

This level pertains to the question of being able to realize a communication between sender and receiver that actually reaches the receiver and is perceptible to him. On the side of the sender this presumes a faultless coding of the message to be sent and a faultless sending of the message to take place. Faultless coding and sending means errorfree instructional software and support software (such as operating system activities involved in the distribution of the courseware). No viable

support tools do as yet exist for the systematic production of error free instructional program code.
Instructional program generators as they exist today are extremely limited in scope and hardly ever lead to adequate courseware. The compiler or interpreter used for the translation of the commands in machine readable code offers a mechanism however for the detection of (syntactic) errors. So the developer of courseware has to some extent a tool available for checking whether the program code of the courseware is accurate.

Level 2. The transmission of the desired meaning

This level pertains to the decodability of the message for the receiver. If the receiver is not able to properly decode the sent message, that is to attach meaning to it, effective communication can not be realized. On the side of the sender this presumes that the message is coded in such a way that semantic correctness during decoding can be realized by the receiver.
Support for the realization of semantic correct courseware is to be found in the availability of methods that mark a developmental path or give a set of working directions by means of which the developer can systematically work from the specifications of a required product towards an actual product, satisfying these specifications. As far as the development of computer programmes is involved one can in this respect think of software engineering methodologies (e.g. Jackson 1985, SADT, Ross 1977).
As far as the development of instructional systems is involved, several instructional design approaches (e.g. Gagne and Briggs 1974, Romiszowski 1982, 1984) are available. If however integrated courseware design is involved one can think of the use of instructional (software) templates that in some way or other pre-structure the courseware to be produced. Using these templates the developer can rely upon preformatted instructional program structures. Quite often, the use of these templates leads to rather rigid and restricted courseware.

Checks on semantic correctness are to be based upon the degree of congruity between the original specifications and the characteristics of the product.

Where clear, unambigue specifications were given at the start of the development process, these checks should not be too difficult. Quite often however, courseware specifications are not that unambigue.

Level 3. Effecting conduct in a desired way

This level pertains to the effect the decoded message has upon the receiver. Effecting conduct in a desired way pertains to the so called pragmatics problem. Pragmatic effectiveness is context dependent, that is the context in which sender and receiver communicate. In order for pragmatic effective communication to take place, on the side of the sender the felicity principle has to prevail. Jackson (1985) speaks about felicity being concerned with whether or not a particular act is appropriate in a particular context.
Felicity in respect to the instructional events (that can be considered to be the instructional acts as far as the sender is involved in computer based learning) is based upon the integrity of the whole of instructional events that make up the courseware. Doblin (1980) speaking about the pre-conditions of the pragmatics of non-textual communications, sees integrity as the right assortment and assemblage of the parts of which a message is compcsed.
On the side of the receiver, several pre-conditions have to be fullfilled in order for pragmatic effectiveness to be realizable as far as computer based learning is involved. The receiver has to be sincere in his/her intention to acquire information, has to accept the sent information and to perceive the message as credible.

For the developer of courseware, it is quite hard to know in advance whether these pre-conditions on the side of the receiver will be met. Given this problem, however, the developer could strive after integrity in his courseware. Integrity in courseware then would pertain to the right assortment and assemblage of instructional events to be offered by the courseware.

No specific software methods are as yet available for the support of realizing effective courseware ss considered on this level. In the educational literature, the term "formative evaluation" is often encountered, meaning evaluation of a product that is under development. In formative evaluation the product that is being developed is tested in a stepwise fashion. For instance a prototype of instructional material is tested on its functionality in a single classroom, revised and tested on school level afterwards. Formative evaluation as usually carried out however is not based on a model of the system under development. Therefore, the developer using regular formative evaluation techniques is not able in a pre-prototype product stage to test the likely effects of various design options and based upon the obtained results to choose the best ones for realizing the prototype.

So we are left in a situation in which it is possible during the authoring of courseware to check the syntactic correctness of the computer code involved, to get some support for realizing semantic correct courseware, but little or no support for the realization of effective courseware during design.

As the developer of courseware is dependent on the functionality and quality of the tools he uses and as these tools appear only to be instrumental as far as testing on syntactic correctness of courseware is involved, the question why the effort involved in the production of courseware quite often does not stand up to the resulting product quality, seems a little more answerable now.

Our present research is an attempt to explore what kind of instructional development methodology and kinds of tools are feasible to support the design of effective courseware (from a pragmatics point of view), given that syntactic and semantic correct courseware can be realized.

Integrity of courseware, forming a part of the pragmatics question, is context dependent. As far as CAI is concerned, broadly two main categories of instructional context (instructional approaches and connected types of instructional goals) are involved. The question "Who is steering the instructional process?" is a paramount divisor herein. System controlled CAI presumes that the computer guides the learner through the instructional material. Typically, the instructional goals involved in this category stress the importance of systematic instructional activities and a strive towards a measurable increase of knowledge to be realized by means of the instructional activities. Furthermore implicit or explicit rules are involved in respect to decision processes for guiding the learner through the instructional material. Such a rule could be " go to the next unit of instructional material only if all questions in this unit are correctly answered".

Representatives of this category are the CAI forms drill-and-practice and tutorial. Drill-and-practice is a stepwise approach aimed at procedural mastery predominantly of sequences of (cognitive) operations. Tutorial is primarily used for the instruction of conceptual or factual relationships and often drill-and-practice is embedded in tutorial CAI.

Learner controlled CAI presumes that the learner is his own guide through the instructional material. The instructional goals in this category often stress the importance of opportunities for the learner to explore the courseware freely. The instructional process involved typically has a nonsystematic character; the learner decides what and how to learn and when he/she has learned enough.

Representatives of this category are the CAI forms simulation and inquiry.
Simulation, based upon a model of a part of the empirical world makes it possible for a learner to explore what the effects are if the parameters of the model are given other values. Inquiry pertains to some forms of retrieving from an electronic "dictionary".

From our perspective there was a choice to be made between the two contexts in terms of our exploration. We have chosen for the context system control, as the degree of control the courseware developer has over the way the instructional process will take place and the instructional events the learner will encounter is much greater here than in the context of learner control. As a consequence of this the developer of system controlled courseware is in a better position to analyze the integrity and therefore the likely effectiveness of his product than the developer of learner controlled courseware.

Even though the two forms of system controlled CAI we have mentioned differ to some extent, we have opted for an approach in which both forms can be used. Tutorial however is the main form from our point of view, under which drill-and-practice can be subsumed.

A workenvironment for analyzing the likely effectiveness of courseware under design, should offer possibilities for analyzing the integrity of the design, for modelling the instructional process and for modelling the learner. By means of posing systematic what-if types of questions, the designer should be able to explore the effects of interactions between design characteristics, the model of the instructional process and the model of the learner. By using quality indicators and by posing criteria that should be met for an allowable design solution, adequate design solutions could be found, given the mentioned facilities of the workenvironment. Furthermore in such a workenvironment modelling of the instructional process and learner should be conform the instructional context choosen. Preferably, the model of the instructional process should be developed as a connected system of adjustable units and the workenvironment should contain a courseware development system, in order to be able to trace its functionality.

So, modelling is the first step. Schemes of tutorial (instruction) systems are given (among others) by Stolurow (1971), Hartley and Sleeman (1975) and Wagner (1981).
Even though these schemes differ in degree of elaboration and to some extent in the functions embodied in them, a common divisor of these schemes was clearly extant. Yet none of these schemes could be used for our exploration as they were not developed for our research objective and miss the necessary degree of formalization.

3. GENERAL SYSTEMS THEORY AS A FRAMEWORK FOR MODELLING

To meet this need of testing already in the developing stage the effects of alternative design choices, the construction of a model is essential. The instructional variables of this model can then be manipulated to test their impact on student performances.

Conceptually, the most simple model is a qualitative model or block diagram consisting of blocks and errors. The blocks refer to functional parts of the CAI-system and the arrows connecting the blocks indicate that there is a relationship of some kind. 'Tutorial schemes' can be considered to belong to this type of models. Block diagrams provide a verbal description of a system by enumerating the important elements and their relationships relative to a given problem (March and Simon 1961).

The qualitative model used in this paper has been constructed to represent an abstract tutorial system as a first step in building a model of the tutorial CAI process, and will later be transformed into an adjustable tutorial software machine.

The teaching material is localized in the subject matter block. The subject matter will be represented by a collection of four so-called elementary instructional frames, in a way to be described later.
It is typical for tutorial CAI that a small piece of the subject matter together with a question is presented to the student represented by the student block. The actual answer given by the student to this question is compared with the right answer stored in the matching block. The result of the matching procedure is sent to the score block.

The score block and the decision block interact with each other by means of a simple feedback mechanism. On the one hand the past history of the student is collected in the score block, which is used in the decision block to decide how to proceed with the instruction. The next frame to present to the student is based on a detailed specification of decision rules depending on the student score. These rules reflect part of the teaching strategy and establish the route the student is going to follow through the network of frames. On the other hand, it is decided in the decision block whether the counter of the student score has to be adjusted or not.

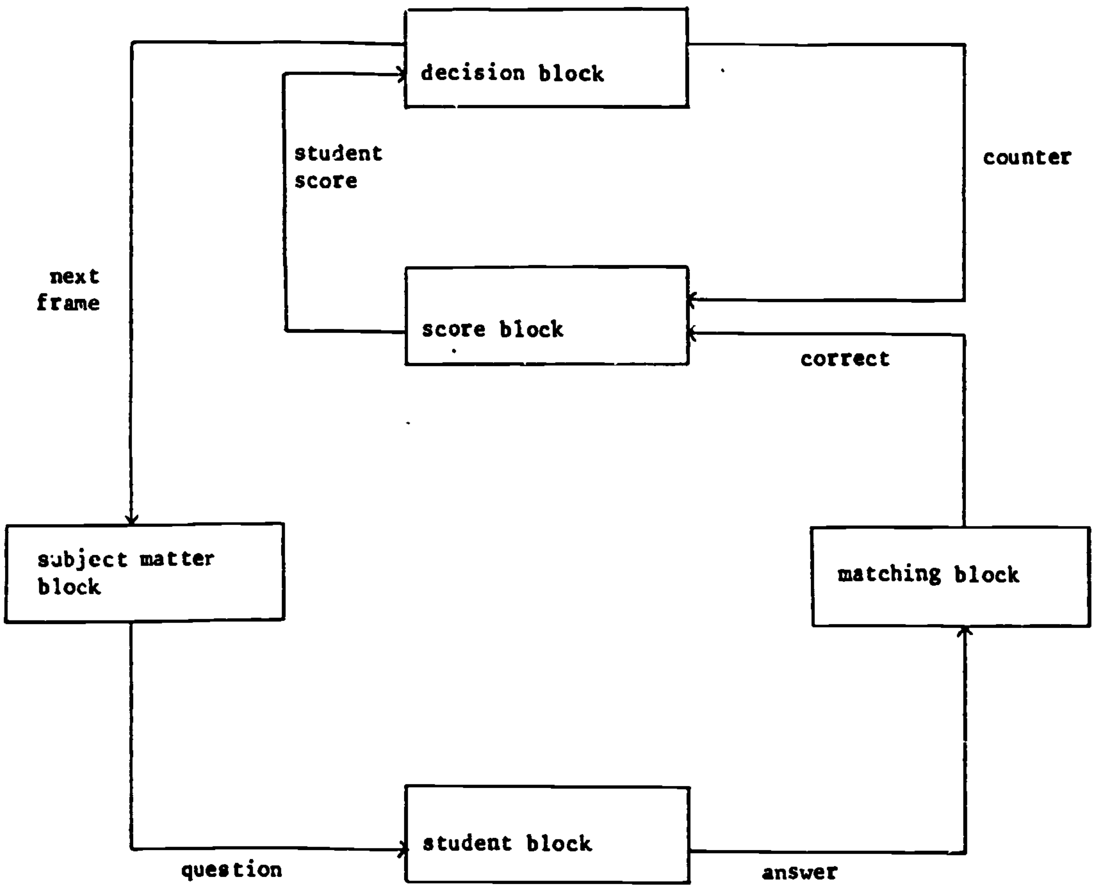

Figure 1 Block diagram of a system controlled CAI-system
blocks: functional parts
arrows: relationships

As indicated before, this qualitative model is inadequate for our purpose, but it can serve as an aid to construct a formal model. A formal model is best suited to describe the dynamic behavior of complex systems with feedback phenomena and many interactions between components (Forrester, 1961). Such a formal model can be used to represent a connected system of adjustable units. The units result from a formal description of the blocks in Figure 1 and they become adjustable by formalizing the relationships.

Once this adjustable tutorial software machine has been created, it can be used for bringing together the right assortment and assemblage of instructional events. In this way the developer can analyze the integrity of the courseware under design with higher chances on developing effective courseware.

To formalize the block diagram the General Systems Theory is used as a framework. Its concepts offer the tools to give a mathematical description of a complex whole of interacting parts (Bertalanffy 1968). In this paper only those concepts will be considered, which are needed to formalize the block diagram. It will be indicated how the General Systems Theory can be used to describe the blocks and arrows in an unambigious way.

4. SOME GENERAL SYSTEMS THEORY NOTIONS

To be able to classify the blocks of Figure 1, some elementary systems will be introduced.
The first elementary system to be considered is the well-known black box (see SC in Figure 2). From this elementary system only the relationships between the independent input variables and the dependent output variables are known. The next elementary system to be considered is the black box with memory. In addition to the input and output variables they possess state variables. Those dependent variables neccessary and sufficient to determine the output variables together with the input variables are called the state variables. They contain the relevant past history and serve as the memory of the system. The last elementary system to be considered is the decision system (see DC in Figure 2). Decisions are taken by means of decision rules being of an 'if then' character, which means that certain prescribed actions are taken if certain conditions are fullfilled.

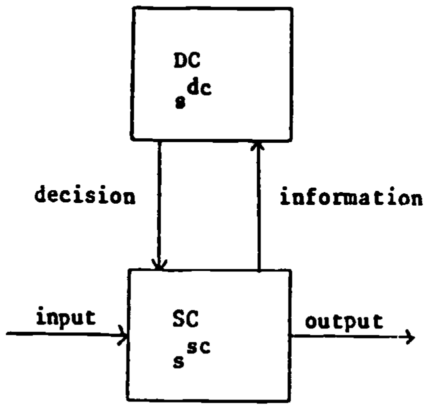

Figure 2 The defined elementary systems and the corresponding system variables
DC: decision system
SC: black box
s^{sc}, s^{dc}: state variables of the black box and decision system

The following symbols will be used to indicate the system variables at time t:

$x(t)$ = input variable;

$y(t)$ = output variable;

$s(t)$ = state variable;

$u(t)$ = decision variable;

$z(t)$ = information variable;

By means of these variables the arrows of Figure 1 can be classified. To describe the relationships between the arrows, it is neccessary to formulate the general form of the two fundamental system equations:

$y(t) = f[x(t),s(t),u(t)]$: output equation ;

$s(t+1) = g[x(t),s(t),u(t)]$: state equation ;

The information and decision equations are special cases of the output equations and have the same general form.

5. THE SUBJECT MATTER REPRESENTED BY A COLLECTION OF FOUR ELEMENTARY FRAMES.

An essential role in all four elementary frames is played by the pointers, of which the types I until III and IV possess respectively 2 and 4. A pointer refers to the next frame to be presented to the student after his/her response. The types I until III build up a hierarchy, by which is meant that the higher ones incorporate minimally all the properties of the lower ones.
It is further assumed that the subject matter is composed of frames, and, in turn every frame of levels. The frames are used to partition the instructional material, for instance, on the basis of task analysis. Every frame belongs to one of the four elementary types. Frames at the same level contain teaching materials of the same difficult, while frames at a higher level con- tain more difficult material. In this way every frame can be indicated unambigiously by a frame and a level-number.

The first elementary frame to be discussed is of type I. It consists of several questions and is specially suited for drill-and-practice purposes. In this type pointers are distinguished. An up-pointer refers to a frame at a higher level, while a down-pointer refers to a frame at the same or a lower level. Not every student-response activates a pointer and in such case the next question in the frame is presented to the student. Whether a pointer is activated or not is decided on the basis of the student-score in the decision block.
In addition to all the properties of type I the second type has the possibility of presenting a preceding text at the start of the first question. Type III is, like type II, a mixture of a pure drill-and-practice and a tutorial frame. Along with all the properties of type II, this frame has the possibility of giving additional information about the chosen alternative. Whether this will be given or not is decided in the decision block again.
The last elementary frame to be discussed is of type IV which is somewhat different from the first 3 types. It is specially suited for pure tutorial purposes. The essential difference with the other 3 types lies in the fact that now a pointer is used to relate every alternative to

the next frame to be presented. In the case of a multiple-choice question with four alternatives this means that the '1','2','3', and '4' responses are related respectively to the 'A','B','C', and 'D'-pointer. Which frame actually is presented to the student depends for all four types on the past history of the student and is controlled by the rules in the decision block. This means that the designer hast to prepare all the frames containing the instructional material as well as all possible routes which link these frames together. Also, the designer has to give a detailed specification of the decision rules. These decision rules can easily be changed by the designer without changing the instructional material. This property makes the CAI-system into a connected system of adjustable units. The machine, in turn, collects the student data and takes the routing decisions on the basis of these data.

6. FORMALIZATION OF THE BLOCK DIAGRAM USING THE GENERAL SYSTEMS THEORY

Having introduced the elementary frames as the tools to conatruct the instructional material, the block diagram of Figure 1 can now be formalized by means of the mentioned concepts of General Systems Theory (G.S.T.).

All the blocks and arrows will be interpreted in terms of both the elementary systems and system variables. As an example, the arrows leaving the matching block, the score block, and the decision block will be described by their system equations. This inplies that, in order to illustrate how the decision block works, some decision rules have to be specified.

The G.S.T.-model can be represented as shown in figure 3.

A few more arrows have been included in the G.S.T.-model to make a complete description possible. It can be noticed from the model that the blocks of Figure 1 are represented by the following elementary systems. The subject matter and the student block respectively by the black boxes SC. sub and SC. stu. The score block by black box SC. sco, which has a memory s^{sco}. The matching block by decision system DC.mtc. And finally, the decision block by decision system DC. dec, which has a memory s^{dec}.

Before starting one cycle through the G.S.T.-model some notations will be introduced. Above the system variable the block of descent will be marked and beneath the block of destination. A system variable can be composed of several components, each indicated by a number. For instance, $u^{mtc}_{sco}(2,t)$ stands for the second component of the decision variable at time t descent from the matching block and with the score block as a destination.

The cycle through the G.S.T.-model starts at the subject matter block SC.sub. On the basis of the decision variable u^{dec}_{sub} the right block, level, and question number is selected by means of a retrieval mechanism. If it is decided in DC. dec to give additional information this is also picked up in SC. sub and presented to the student. The listing of the system variables and equations is attacked as an Appendix with a brief explanation.

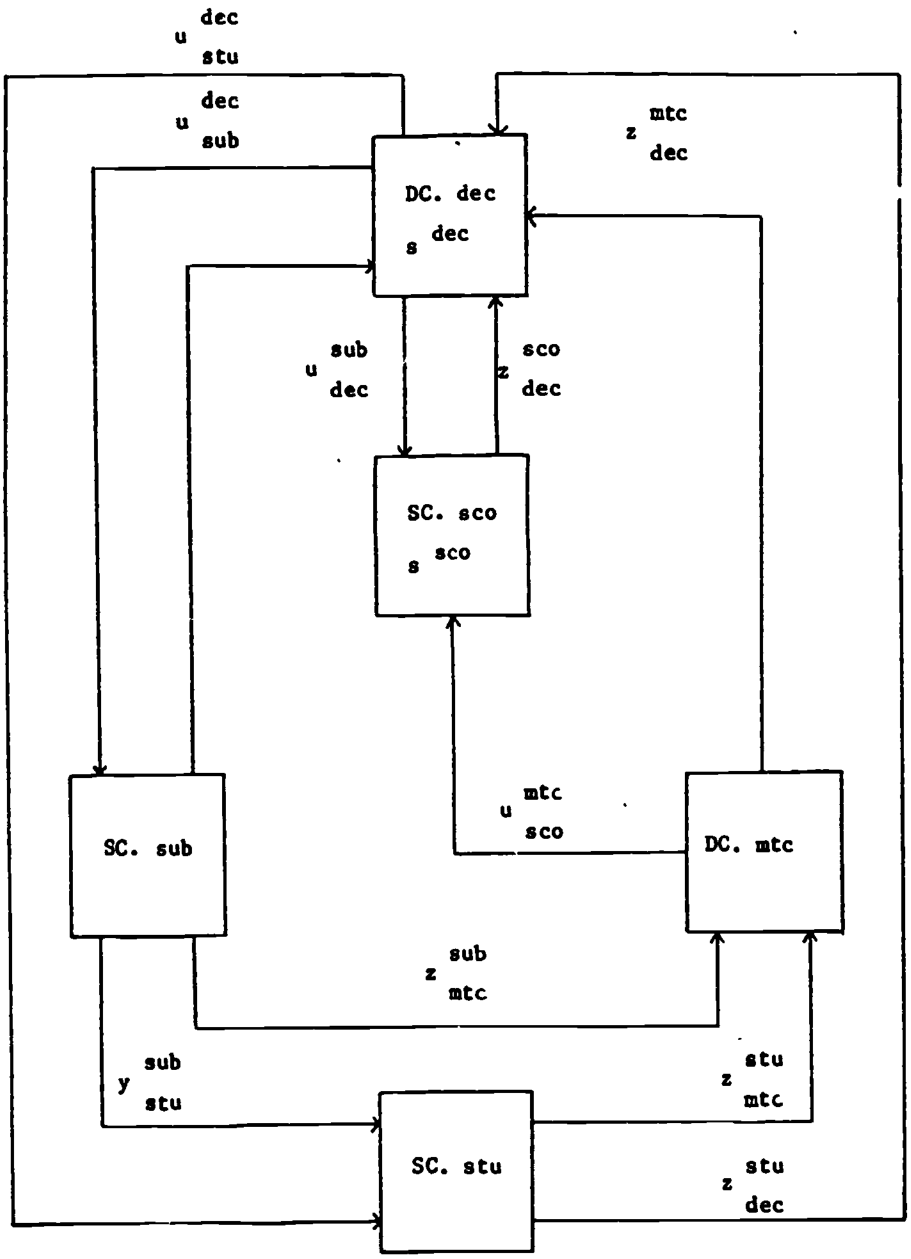

Figure 3 A G.S.T.-model of a system controlled CAI-system

Once the system equations have been written down, the G.S.T.-model can easily be converted into a computer program. The main program can be written by developing procedures in a modular way for each elementary system. The input variables appear as value-parameters and the output, decision, and information variables as var-parameters in the procedure body. The described system equations are used in the procedure body. Before being able to run the main program it is necessary to initialize the components of some system variables.

The main program is running now in the following way:

```
BEGIN
  initialize (student-decision, student-history);
WHILE NOT student-decision.terminate DO
  BEGIN
  subject-matter (student-decision, block);
  readln (input, answer);
  matching (frame, answer, matching-result);
  score (matching-result, student-decision, student-history);
  decision (student-history, frame, student-decision);
  END
END.
```

This computer program, developed by transforming the G.S.T.-model into a set of cooperating procedures, can be conceived as the adjustable tu-torial software machine introduced in section 1.

7. CONCLUSIONS

The development of analytical tools for the developer of courseware is without any doubt a very promising perspective. Yet the road to go is not sn obvious one nor an easy one to walk on.
The presented research work has demonstrated that it is possible to define unambigiously a tutorial process and to develop an execution mechanism for courseware that is both manipulable and based upon a modular and explicite mathematicel model.
This model if integrated in a workenvironment for courseware, can be used in connection with analytical tools to trace the likely effects of varioue design options. As far as the presented research is concerned, only the choices and effects of dicusionrules in the context of system controlled CAI are manipulsble an traceable. The presented model is a rather simple one though it may seem complicated. Rather, to cover tutorial processes in full plurifold, it would take a fair extension of the presented model and a more general approach still to the construction of system cells. Research in this directions is carried out by De Diana end Vos. Furthermore, in the presented research no specific analytical tools have baen described. Such tools are under development in the CAD-CAI project, forming part of an experimental work environment for the development of coursweware (called EDUC). Further extensions of the presented research can be expected from studying the contect of learner controlled CAI and the addition of a "memory cel" for the re-presentation of the shown behaviour of the learner during the instructional process.

REFERENCES

AVNER, R. (1979), Production of computer-based instructional materials. In: O' Neill (Ed.): Issues in instructional Systems Development. Academic Press, New York.

BARKER, P., SINGH, R. (1982), Authoring languages for Computer-Based Learning. Brit. Journal of Educ. Technology, no. 3, vol. 13, 167-196.

BERTALANFFY, L. von. General Systems Theory. New York, 1968.

BORK, A. (1981), Learning with computers. Digital Press, Bedford, Mass.

BORK, A. (1984), Producing Computer Based Learning materials at the Educational Technology Center. Journal of Computer Based-Instruction, vol. 11, no.3, 78-81.

COBLIN, J. (1980), A structure for nontextual communications. In : Kolers, P., Wrolstad, M., Bouma, H. (eds.) The processing of visible language 2. Plenum Press, New York.

FAIRWEATHER, P., O'NEAL, A. (1984), The impact of advanced authoring systems on CAI productivity. Journal of Computer Based-Instruction, vol. 11, no.3, 90-94.

FORRESTER, J.W. (1961) Industrial Dynamics. Cambridge (Mass.).

FRANCIS, L. (1979), Five phases in the life of CBT sites: II. Staf selection and retention. ADCIS Proceedings, San Diego, Calif.

GAGNE, R., BRIGGS, L. (1974), Principles of Instructional Design. Holt, Rinehart and Winston, New York.

HARTLEY, J., SLEEMAN, D. (1975), Towards More Intelligent Teaching Systems. Int. J. Man-Machine Studies, 5, 215-236.

JACKSON, M.A. (1975), Principles of program design. Academic Press.

JACKSON, P. (1985), Reasoning and advice-giving systems, 73-83 in: Bramer, M. (ed.) Research and development in expert systems. Cambridge University Press, Cambridge.

KEARSLEY, G. (1982), Authoring systems in Computer Based Education. Communications of the ACM, 429-437.

KEARSLEY, G., HILLELSOHN, M. (1982), Human factors considerations for CT systems. Journal of Computer Based Instruction.

KEARSLEY, G. (1983), Computer Based Training. Addison & Wesley, Reading.

KONTOS, G. (1984-1985), Instructional computing: in search of better methods for the production of CAI lessons. J. Educ. Technology Systems, vol 13, 1.

MARCH, J.G., SIMON, H.A. (1961) Organizations. New York.

MOONEN, J., GASTKEMPER, F. (1983), Computer gestuurd onderwijs. Het Spectrum, Utrecht.

ROMISZOWSKI, A. (1982), Designing Instructional Systems. Kogan Page, London.

ROMISZOWSKI, A. (1984), Producing instructional Systems. Kogan Page, London.

ROSS, D.T. (1977), Structured Analysis: a Language for communicating Ideas. IEEE trans. on software engineering, vol. 3, no.1, 16-34.

SHANNON, C., WEAVER, W. (1949), The mathematical theory of communication. University of Illinois Press, Urbana.

STOLUROW, L. (1971), Models for instructional design: a systems approach to instruction. In : Merrill, M.D. (Ed.): Instructional Design: Readings. Prentice Hall, Englewood Cliffs.

WAGNER, W. (1981-1982), Design considerations for instructional computing programs. J. Educ. Technology Systems, vol. 10, 3.

APPENDIX

The listing of the system variables and equations of the G.S.T.-model.

We start with a description of the output and information variables leaving the subject matter block.

$y^{sub}_{stu}(1,t) =$	presented question and its accompanying alternatives, possibly preceding by a text in case of type II until IV;
$y^{sub}_{stu}(2,t) =$	presented additional information if decided in case of type III and IV;
$z^{sub}_{mtc}(1,t) =$	number of correct alternative (1,2,3 or 4);
$z^{sub}_{mtc}(1+i,t) =$	alternative information belongs to the ith alternative (boolean var.); $1 < i < 4$
$z^{sub}_{dec}(1,t) =$	selected block belongs to type IV (boolean var.);

For types I until III:

$z^{sub}_{dec}(2,t) =$	number of questions in selected frame;
$z^{sub}_{dec}(3,t) =$	frame number assigned to the up-pointer;
$z^{sub}_{dec}(4,t) =$	level number assigned to the up-pointer;

$z^{sub}_{dec}(5,t)$ = frame number assigned to the down-pointer;

$z^{sub}_{dec}(6,t)$ = level number assigned to the down-pointer;

For type IV:

$z^{sub}_{dec}(7,t)$ until $z^{sub}_{dec}(10,t)$ = frame numbers assigned to respectively the 'A', 'B', 'C' and 'D'-pointers;

$z^{sub}_{dec}(11,t)$ until $z^{sub}_{dec}(14,t)$ = level numbers assigned to respectively the 'A', 'B', 'C' and 'D'-pointers;

Information variables leaving the student block:

$z^{stu}_{mtc}(1,t) = z^{stu}_{dec}(1,t)$ = student answer (1,2,3 or 4);

Decision and information variables leaving the matching block:

$u^{mtc}_{sco}(1,t)$ = the student answer is correct (boolean var.);

$z^{mtc}_{dec}(1,t)$ = additional information belongs to the student answer (boolean var.);

Components of the information variable $z^{sco}_{dec}(t)$ leaving the score block coincide with components of the state variable s^{sco}:

$z^{sco}_{dec}(1,t) = s^{sco}(1,t)$ = number of correct answers in actual frame;

$z^{sco}_{dec}(2,t) = s^{sco}(2,t)$ = total number of questions which have been asked until time t;

$z^{sco}_{dec}(3,t)[i,j] = s^{sco}(3,t)[i,j]$ = number of times the frame with frame number i and level number j has been visited.

The last block to be discussed is the decision block, where we start with a description of the components of the state variable $s^{dec}(t)$:

$s^{dec}(1,t)$ until $s^{dec}(3,t)$ = respectively frame, level and question number of the actual frame.

The first component of the decision variable $u^{dec}_{sub}(t)$ is declared as a defined type in PASCAL:

```
u^dec_sub (1,t) = switch = (down,     (*decrease level or stay at the same
                                       level*)
                    up,               (*increase level*)
                    next,             (*present next question in the frame*)
                    not-used,         (* frame belongs to type IV *));
```

$u^{dec}_{sub}(2,t)$, $u^{dec}_{sub}(3,t)$ and $u^{dec}_{sub}(4,t)$ = respectively frame, level and question number of the possible new frame to present;

$u^{dec}_{sub}(5,t)$ = terminate the instruction because of sufficient mastery (boolean var.);

$u^{dec}_{sub}(7,t)$ = terminate the instruction because of exceeding the time (boolean var.);

$u^{dec}_{sub}(8,t)$ = terminate the instruction because of sufficient mastery, insufficient mastery or exceeding the time (boolean var.);

$u^{dec}_{sub}(9,t)$ = additional information is presented (boolean var.);

$u^{dec}_{sub}(10,t)$ until $u^{dec}_{sub}(13,t)$ = respectively the frame, level, question and alternative number of the additional information in case of presenting.

The components of the decision variable with destination the scoreblock

$u^{dec}_{sco}(1,t)$, $u^{dec}_{sco}(2,t)$, $u^{dec}_{sco}(3,t)$ and $u^{dec}_{sco}(4,t)$, coincide with

$u^{dec}_{sub}(1,t)$, $u^{dec}_{sub}(2,t)$, $u^{dec}_{sub}(3,t)$ and $u^{dec}_{sub}(9,t)$, respectively.

Component of the decision variable with destination the student block:

$u^{dec}_{stu}(1,t)$ = presented text in case of stopping;

Finally, the subject matter block is reached again and one complete cycle through the G.S.T.-model has been completed. If the decision rules have not decided to stop, i.e.

$u^{dec}_{sub}(8,t)$ = false, this is the starting point for a new cycle.

To illustrate the use of the system eqations some of them will be discussed now, starting with the matching block:

$$u^{mtc}_{sco}(l,t) = \text{true, if } z^{stu}_{mtc}(l,t) = z^{sub}_{mtc}(l,t) \text{ ;}$$

$$z^{mtc}_{dec}(l,t) = \text{true, if } z^{sub}_{mtc}(l+i,t) = \text{true and } l+i = z^{stu}_{mtc}(l,t);\ (1 < i < 4)$$

The next block to be discussed is the score block, starting with the state equations:

$$s^{sco}(l,t+1) = s^{sco}_{sco}(l,t) + 1, \text{ if } u^{dec}_{sco}(l,t) = \text{next and } u^{mtc}_{sco}(l,t) = \text{true;}$$

$$= s^{sco}(l,t) \quad \text{, if } u^{dec}_{sco}(l,t) = \text{next and } u^{mte}_{sco}(l,t) = \text{false;}$$

$$= 1 \quad \text{, if } u^{dec}_{sco}(l,t) = \text{(up or down or not-used)}$$

$$\text{and } u^{mtc}_{sco}(l,t) = \text{true;}$$

$$= 0 \quad \text{, if } u^{dec}_{sco}(l,t) = \text{(up or down or not-used)}$$

and $u\,\overset{\text{mtc}}{\underset{\text{sco}}{(1,t)}} = \text{false};$

$\overset{\text{sco}}{s}\,(2,t+1) = \overset{\text{sco}}{s}(2,t) + 1$;

$\overset{\text{sco}}{s}\,(3,t+1)\,[i,j] = \overset{\text{sco}}{s}\,(3,t)\,[i,j] + 1$, if $u\,\overset{\text{dec}}{\underset{\text{sco}}{(2,t)}} = i$ and $u\,\overset{\text{dec}}{\underset{\text{sco}}{(3,t)}} = j$ and

($u\,\overset{\text{dec}}{\underset{\text{sco}}{(1,t)}}$ = up or down or not-used)

and $u\,\overset{\text{dec}}{\underset{\text{sco}}{(4,t)}} = \text{false};$

$= \overset{\text{sco}}{s}\,(3,t)\,[i,j]$, if $u\,\overset{\text{dec}}{\underset{\text{sco}}{(z,t)}} = i$ and $u\,\overset{\text{dec}}{\underset{\text{sco}}{(3,t)}} = j$ and

$u\,\overset{\text{dec}}{\underset{\text{sco}}{(1,t)}}$ = next and $u\,\overset{\text{dec}}{\underset{\text{sco}}{(4,t)}} = \text{false};$

$= 1$, if $u\,\overset{\text{dec}}{\underset{\text{sco}}{(2,t)}} = i$ and $u\,\overset{\text{dec}}{\underset{\text{sco}}{(3,t)}} = j$ and

($u\,\overset{\text{dec}}{\underset{\text{sco}}{(1,t)}}$ = up or down or not-used) and

$u\,\overset{\text{dec}}{\underset{\text{sco}}{(4,t)}} = \text{true};$

$= 0$, if $u^{dec}_{sco}(2,t) = i$ and $u^{dec}_{sco}(3,t) = j$ and

$u^{dec}_{sco}(1,t) = \text{next}$ and $u^{dec}_{sco}(4,t) = \text{true}$;

The three information equations for $z^{sco}_{dec}(1,t)$, $z^{sco}_{dec}(2,t)$ and $z^{sco}_{dec}(3,t)$ [i,j]

coincide with the state equations for

$s^{sco}(1,t)$, $s^{sco}(2,t)$ and $s^{sco}(3,t)$ [i,j], respectively.

The last block to be discussed is the decision block, starting with the decision equations with respect to the subject matter block:

$u^{dec}_{sub}(1,t) = \text{up}$, if $z^{sco}_{dec}(1,t) > \text{round}\ (0{,}6 \times z^{sub}_{dec}(2,t)$ and $z^{sub}_{dec}(1,t) = \text{false}$;

$= \text{down}$, if $s^{dec}(3,t) > \text{round}\ (0{,}8 \times z^{sub}_{dec}(2,t)$ and

$z^{sco}_{dec}(1,t) < \text{round}\ (0{,}4 \times z^{sub}_{dec}(z,t))$ and $z^{sub}_{dec}(1,t) = \text{false}$;

$= \text{not-used}$, if $z^{sub}_{dec}(1,t) = \text{true}$;
$= \text{next}$, else;

In words, the specified decision rules for the types I until III run as follows:
The up-pointer is activated if more than 60% of questions in the frame

have been answerel correctly. The down-pointer is activated either if more than 80% of the number of questions in the frame have been answered and less than 40% of them have been answered correctly or if all the questions in the frame have been answered. In all other cases the next question in the frame is presented.

$u^{dec}_{sub}(2,t) = s^{dec}(1,t)$ and $u^{dec}_{sub}(3,t) = s^{dec}(2,t)$, if $u^{dec}_{sub}(1,t)$ = next;

$u^{dec}_{sub}(2,t) = z^{sub}_{dec}(3,t)$ and $u^{dec}_{sub}(3,t) = z^{sub}_{dec}(4,t)$, if $u^{dec}_{sub}(1,t)$ = up;

$u^{dec}_{sub}(2,t) = z^{sub}_{dec}(5,t)$ and $u^{dec}_{sub}(3,t) = z^{sub}_{dec}(6,t)$, if $u^{dec}_{sub}(1,t)$ = down;

$u^{dec}_{sub}(2,t) = z^{sub}_{dec}(7 + i,t)$ and $u^{dec}_{sub}(3,t) = z^{sub}_{dec}(11 + i,t)$, if $u^{dec}_{sub}(1,t)$ = not used

and $z^{stu}_{dec}(1,t) = i$; $(1 < i < 4)$.

$u^{dec}_{sub}(4,t) = s^{dec}(3,t) + 1$, if $u^{dec}_{sub}(1,t)$ = next;

$= 1$, if $u^{dec}_{sub}(1,t)$ = up, down or not-used;

$u^{dec}_{sub}(5,t)$ = true, if $u^{dec}_{sub}(3,t)$ = maximum level + 1;

$u^{dec}_{sub}(6,t)$ = true, if $u^{dec}_{sub}(3,t)$ = minimum level -1;

$u^{dec}_{sub}(7,t)$ = true, if $z^{sco}_{dec}(2,t)$ = maximum number of permitted questions;

The maximum level, minimum level and maximum number permitted questions in our example are put respectively on 3,1 and 50.

$u^{dec}_{sub}(8,t)$ = true, if $u^{dec}_{sub}(5,t)$, $u^{dec}_{sub}(6,t)$ or $u^{dec}_{sub}(7,t)$ = true;

$u^{dec}_{sub}(9,t)$ = true, if $z^{sco}_{dec}(3,t)\cdot[i,j]$ = 2 and $u^{dec}_{sub}(2,t)$ = i and

$u^{dec}_{sub}(3,t)$ = j and $z^{atc}_{dec}(1,t)$ = true;

In words, this last decision rule runs as follows: Additional information is presented only each second time a student arrives at a frame which actually contains additional information.

$u^{dec}_{sub}(10,t) = z^{dec}(1,t)$ and $u^{dec}_{sub}(11,t)$ and $u^{dec}_{sub}(12,t) = z^{dec}(3,t)$

and $u^{dec}_{sub}(13,t) = z^{stu}_{dec}(1,t)$, if $u^{dec}_{sub}(9,t)$ = true;

The decision equations with respect to the score block for

$u^{dec}_{sco}(1,t)$ until $u^{dec}_{sco}(4,t)$ coincide with the decision equations for

$u^{dec}_{sub}(1,t)$, $u^{dec}_{sub}(2,t)$, $u^{dec}_{sub}(3,t)$ and $u^{dec}_{sub}(9,t)$, respectively.

Decision equations with respect to the student block:

$$u^{dec}_{stu}(1,t) = \text{print ('sufficient mastery')}, \quad \text{if } u^{dec}_{sub}(5,t) = \text{true};$$

$$= \text{print ('insufficient mastery')}, \quad \text{if } u^{dec}_{sub}(6,t) = \text{true};$$

$$= \text{print (' you are too long busy')}, \quad \text{if } u^{dec}_{sub}(7,t) = \text{true};$$

The three state equations of the decision block for

$s^{dec}(1,t)$, $s^{dec}(z,t)$ and $s^{dec}(3,t)$ coincide with the decision equations for

$u^{dec}_{sub}(2,t)$, $u^{dec}_{sub}(3,t)$ and $u^{dec}_{sub}(4,t)$, respectively.

A publication by
the Department of Education
of the University of Twente,
P.O. Box 217,
7500 AE Enschede,
The Netherlands

Toegepaste Onderwijskunde